KAIZEN

Setup Spark Development Environment – IntelliJ and Scala

APRIL 21, 2018 By Koushik M L N (https://kaizen.itversity.com/author/koushik/)

As part of this blog post we will see detailed instructions about setting up development environment for Spark and Hadoop application development using Windows.

- We have used Windows 10 for this demo using 64 bit version
- Setup development environment on Windows
- For each of the section we will see
  - Why we need to perform the step?
  - How to perform the step?
  - How we can validate whether it is working as expected?
- We will also develop few programs to validate whether our setup is progressing as expected or not
- In case you run into any issues, please log those in our forums (http://discuss.itversity.com)
- Click here (http://discuss.itversity.com/c/banners) for the coupons for our content. Our training approach is certification oriented.
- Click here (https://labs.itversity.com) to go to our state of the art lab to practice Spark hands on for more realistic experience

Setup Development environment on Windows

We are considering fresh Windows laptop. We will start with Java/JDK on Windows laptop and we will go through step by step instructions to setup Scala, sbt, WinUtils etc.

- For integrated development using IntelliJ
- Typically programming will be done with IDEs such as IntelliJ
- IDEs are typically integrated with other tools such as git which is code versioning tool. Tools like git facilitate team development.
- sbt is build tool for Scala. Once applications are developed using IDE, they are typically built using tools like sbt
- WinUtils is required for HDFS APIs to work on Windows laptop

> *Unless java is setup and validated successfully do not go further. If you need our support, please log the issues in our forums (http://discuss.itversity.com).*

Setup Spark Development Environment on Windows - Introducti…

Setup Java and JDK

- Before getting started check whether Java and JDK are installed or not
  - Launch command prompt – Go to search bar on windows laptop, type **cmd** and hit enter
  - Type `java -version` If it return version, check whether 1.8 or not. It is better to have 1.8 version. If you have other version, consider uninstall and install 1.8 (Search for programs installed and uninstall Java)
  - Type `javac -version` If it return version, check whether 1.8 or not. It is better to have 1.8 version. If you have other version, consider uninstall and install 1.8 (Search for programs installed and uninstall Java)
  - If you need other versions, make sure environment variables point to 1.8
  - If you do not have Java at all, make sure to follow the instructions and

install 1.8 version of JRE and JDK.

- **Why do we need to install Java and JDK?** Scala, Spark and many other technologies require Java and JDK to develop and build the applications. Scala is JVM based programming language.
- **How to install Java and JDK?**
  - Go to official page of Oracle (http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html) where downloads are available
  - Accept the terms and download 64 bit version
- **How to validate?**
  - Use `java -version` and `javac -version` commands in command prompt and see they return 1.8 or not

Setup Scala with IntelliJ

- Now install IntelliJ
- There are 2 versions of IntelliJ **community edition** and **enterprise edition**
- Community edition is free and at times you need to install additional plugins
- Enterprise edition is paid and supported and comes with most of the important plugins pre-installed. Also set of plugins are bundled together as part of enterprise edition
- Unless you have corporate license for now consider installing community edition.
- **Why IntelliJ**?
  - IntelliJ is created by JetBrains and it is very popular in building IDEs which boost productivity in team development
  - Scala and SBT can be added as plugins using IntelliJ
  - Most commonly used tools such as git comes out of the box for versioning the code in the process of application development by teams.
- **How to Install?**
  - Go to the downloads (https://www.jetbrains.com/idea/download) page and make sure right version is chosen.
  - Once downloaded, just double click on installable and follow typical

installation process

- **How to validate?**
  - We will develop a program as part of next section to validate.

Develop Hello World Program

We will see how to create first program using Scala as sbt project.

- Click on New Project
- For the first time, it selects java by default. Make sure to choose Scala and then sbt
- Give name to the project -> **spark2demo**
- Choose right version of Scala -> **2.11.12**
- Choose right version of sbt -> **0.13**

> *It will take some time to setup the project*

Once done you will see

- src directory with the structure **src/main/scala**
- src/main/scala is base directory for scala code
- build.sbt under project
  - **name** – name of the project
  - **version** – project version (0.1)
  - **scalaVersion** – scala version (2.11.12)

```
name := "spark2demo"

version := "0.1"
```

```
scalaVersion := "2.11.12"
```

- Steps to develop HelloWorld program
  - **Right click** on **src/main/scala**
  - Choose **Scala Class**
  - Give name as **Hello World** and **change type to object**
  - Replace the code with below code

```
object HelloWorld {

  def main(args: Array[String]): Unit = {
    println("Hello  World")
  }

}
```

  - Right click and run the program
  - You should see **Hello World** in the console

> *Make sure IntelliJ setup with Scala is done and validated by running Hello World program. In case of any issues, please log in our forums (http://discuss.itversity.com).*

Setup sbt and run application

Once the application is developed, we need to build jar file and migrate to higher environments. sbt is the build tool which is typically used for Scala based projects.

- **Why sbt?**
  - To build scala based applications to jar file
  - Validate jar file to make sure program is running fine
- **How to setup sbt?**

  - Setup sbt by downloading relevant downloadable from this [link (https://www.scala-sbt.org/download.html)](https://www.scala-sbt.org/download.html)
  - For Windows use Microsoft Installer (msi)
- **How to validate sbt?**
  - Copy the path by right clicking the project in IntelliJ
  - Go to command prompt and cd to the path
  - Check the directory structure, you should see
    - src directory
    - build.sbt
  - Run `sbt package`
  - It will build jar file and you will see the path
  - Run program by using `sbt run` command
  - You should see **Hello World** printed on the console

Add Spark dependencies to the application

As we are done with validating IntelliJ, Scala and sbt by developing and running the program, now we are ready to integrate Spark and start developing Scala based applications using Spark APIs.

- Update build.sbt by adding

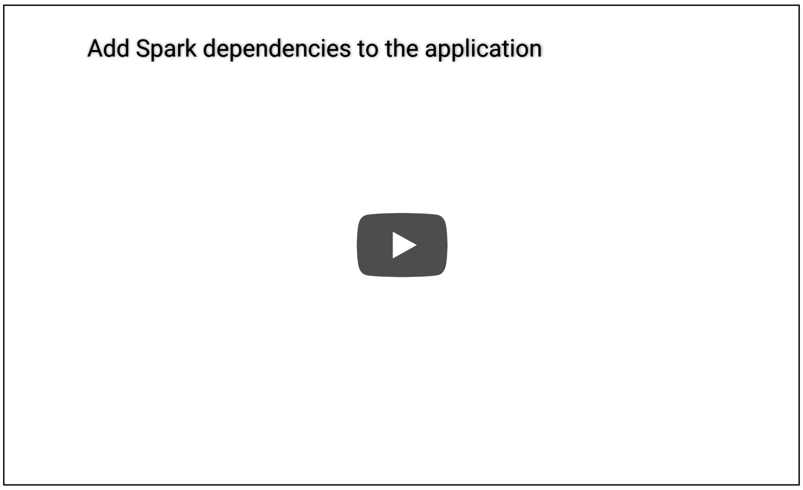
```
libraryDependencies += "org.apache.spark" %% "spark-core" % "2.3.0"
```

- Enable auto-import or click on refresh on type right corner
- It will take some time to download dependencies based on your internet speed

> *Be patient until all the spark based dependencies are downloads. You can expand External Dependencies in project view to see list of jars downloaded.*

- build.sbt will look like this

```
name := "spark2demo"

version := "0.1"

scalaVersion := "2.11.12"

libraryDependencies += "org.apache.spark" %% "spark-core" % "2.3.0"
```



Setup WinUtils to get HDFS APIs working

- **Why to install winutils?**
    - In the process of building data processing applications using Spark, we need to read data from files
    - Spark uses HDFS API to read files from several file systems like HDFS, s3, local etc
    - For HDFS APIs to work on Windows, we need to have WinUtils
- **How to install winutils?**
    - Click here (https://codeload.github.com/gvreddy1210/64bit/zip/master) to download 64 bit winutils.exe
    - Create directory structure like this `C:/hadoop/bin`
    - Setup new environment variable HADOOP_HOME
        - Search for **Environment Variables** on Windows search bar
        - Click on **Add Environment Variables**
        - There will be 2 categories of environment variables
            - User Variables on top
            - **System Variables** on **bottom**
            - Make sure to click on Add for **System Variables**
            - Name: HADOOP_HOME
            - Value: **C:\hadoop** (don't include bin)
        - Also choose **Path** and **click on Edit**
            - Click on Add

- Click on Add
- Add new entry **%HADOOP_HOME%\bin**

Setup Data sets

You need to have data sets setup for your practice.

- Go to our GitHub data repository (https://github.com/dgadirau/data)
- You can setup data sets in 2 ways
  - If you have git, you can clone to the desired directory on your PC
  - Otherwise use download, it will download zip file
    - Unzip and copy to **C:\data**
- You will have multiple datasets ready for your practice

Develop first spark application

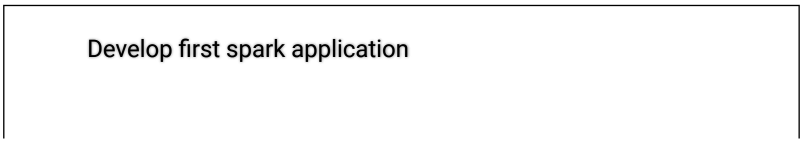
Now we are ready to develop our first Spark application.

- Go to **src/main/scala**
- Right click and click on **New** -> **Package**
- Give the package name as **retail_db**

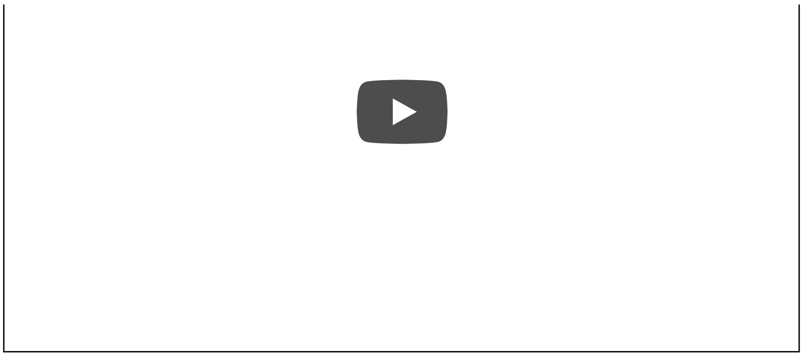
- Right click on retail_db and click on **New** -> **Scala Class**
  - Name: GetRevenuePerOrder
  - Type: Object
- Replace the code with this code snippet

```
package retail_db

import org.apache.spark.{SparkConf, SparkContext}

object GetRevenuePerOrder {
  def main(args: Array[String]): Unit = {
    val conf = new SparkConf().
      setMaster(args(0)).
      setAppName("Get revenue per order")
    val sc = new SparkContext(conf)
    sc.setLogLevel("ERROR")

    val orderItems = sc.textFile(args(1))
    val revenuePerOrder = orderItems.
      map(oi => (oi.split(",")(1).toInt, oi.split(",")(4).toFloat)).
      reduceByKey(_ + _).
      map(oi => oi._1 + "," + oi._2)

    revenuePerOrder.saveAsTextFile(args(2))
  }

}
```

- Program takes 3 arguments
  - args(0) -> execution mode
  - args(1) -> input path
  - args(2) -> output path
- Running the application
  - Go to Run menu -> Edit Configurations
  - Add new application
  - Give application name **GetRevenuePerOrder**
  - Choose main class: **retail_db.GetRevenuePerOrder**
  - Program arguments: **local <input_path> <output_path>**
  - Use classpath for module: Choose **spark2demo**
  - Click on Apply and then Ok
- Now you can run the application by right clicking and choosing **Run “GetRevenuePerOrder”**
- Go to output path and check files are created for output or not



Develop first spark application



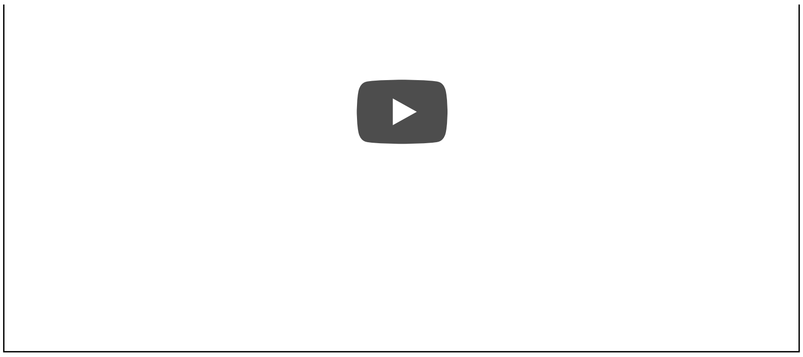
Build jar file

Let us see how we can build the jar file and run it.

- Copy the path by right clicking the project in IntelliJ
- Go to command prompt and cd to the path
- Check the directory structure, you should see
  - src directory
  - build.sbt
- Run `sbt package`
- It will build jar file and you will see the path
- It will be typically <project_directory>/target/scala-2.11/spark2demo_2.11-0.1.jar
- We can also run using sbt “run-main”

```
sbt "run-main retail_db.GetRevenuePerOrder local <input_path> <output_path>"
```

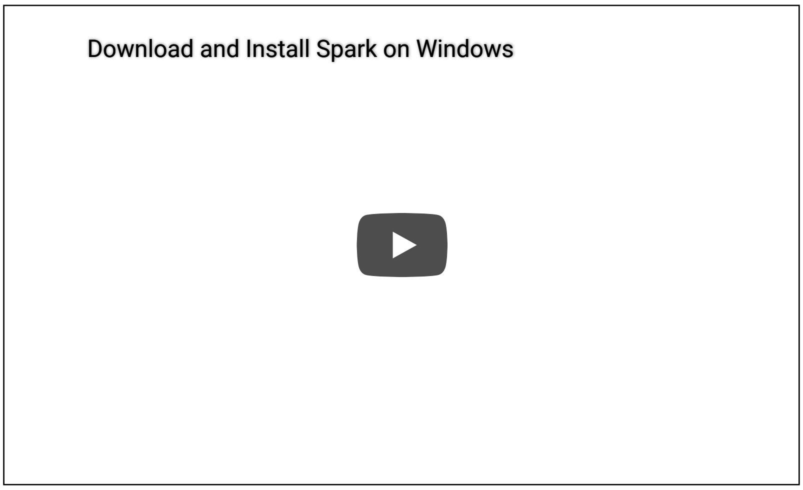
> *Now you are ready with the jar file to get deployed. If you have any issues please raise it in our [forums (http://discuss.itversity.com/c/big-data/apache-spark)](http://discuss.itversity.com/c/big-data/apache-spark).*



Download and Install Spark on Windows

Now let us see the details about setting up Spark on Windows

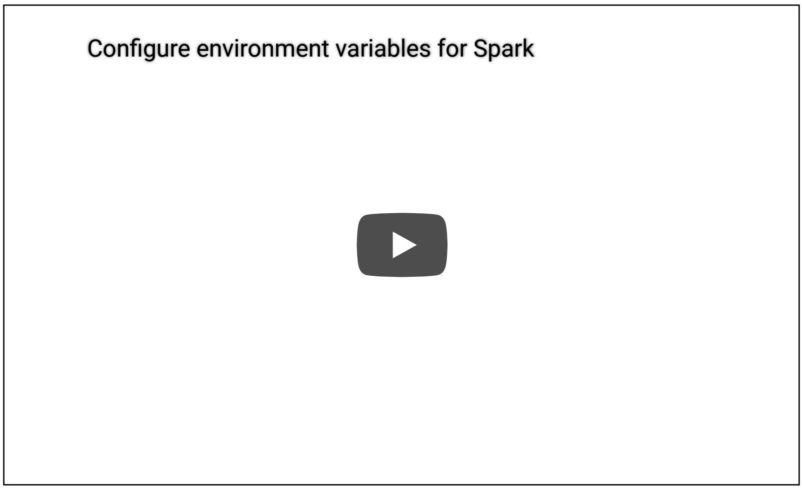
- **Why to setup Spark?**
  - Before deploying on the cluster, it is good practice to test the script using spark-submit.
  - To run using spark-submit locally, it is nice to setup Spark on Windows
- **How to setup Spark?**
  - Install 7z so that we can unzip and untar spark tar ball, from here (https://www.7-zip.org/download.html)
  - Download spark 2.3 tar ball by going here (https://spark.apache.org/downloads.html)
    - Choose Spark Release: **2.3.0**
    - Choose a package type: **Pre-built for Hadoop 2.7 or later**
    - It gives the appropriate link pointing to mirror
    - Click on it go to mirror and click on it to download
    - Use 7z software to unzip and under to complete setup of spark
  - We need to configure environment variables to run Spark any where
  - Keep in mind that Spark is not very well supported on Windows and we will see how to setup on Ubuntu using Windows subsystem for Linux.



Configure environment variables for Spark

Let us see how we can configure environment variables of Spark

- **Why to setup Environment Variables?** To run spark-submit, spark-shell from any where on the PC using the jar file.
- **How to configure Environment Variables?**
  - Let us assume that Spark is setup under **C:\spark-2.3.0-bin-hadoop2.7**
  - Setup new environment variable SPARK_HOME
    - Search for **Environment Variables** on Windows search bar
    - Click on **Add Environment Variables**
    - There will be 2 categories of environment variables
      - User Variables on top
      - **System Variables** on **bottom**
      - Make sure to click on Add for **System Variables**
      - Name: SPARK_HOME
      - Value: **C:\spark-2.3.0-bin-hadoop2.7** (don't include bin)
    - Also choose **Path** and **click on Edit**
      - Click on Add
      - Add new entry **%SPARK_HOME%\bin**
- **How to validate?**
  - Go to any directory and run `spark-shell`



Run Spark job using spark-shell

Using spark-shell we can validate ad hoc code to confirm it is working. It will also confirm whether the installation is successful or not.

- Run `spark-shell`
- Execute this code and make sure it return results

```
val orderItems = sc.textFile("C:\\data\\retail_db\\order_items")
val revenuePerOrder = orderItems.
  map(oi => (oi.split(",")(1).toInt, oi.split(",")(4).toFloat)).
  reduceByKey(_ + _).
  map(oi => oi._1 + "," + oi._2)
revenuePerOrder.take(10).foreach(println)
```

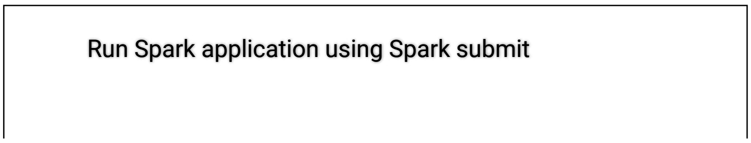
On Windows after showing the output, it might throw the exception.

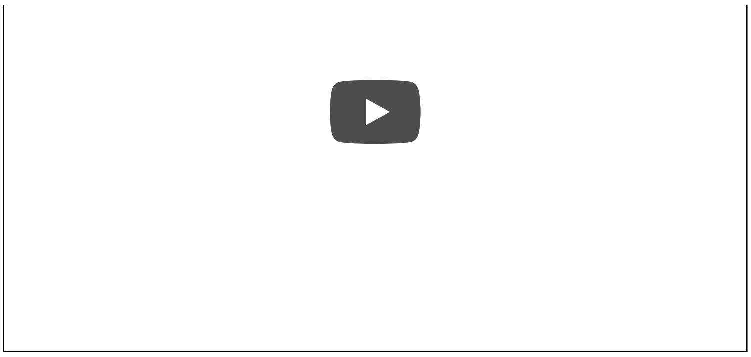
Run Spark application using Spark submit

We can validate the jar file by using spark-submit

- `spark-submit` is the main command to submit the job
- `--class retail_db.GetRevenuePerOrder`, to pass the class name
- By default master is local, if you want to override we can use `--master`
- After spark-submit and control arguments we have to give jar file name followed by arguments

```
spark-submit --class retail_db.GetRevenuePerOrder <PATH_TO_JAR> local <INPUT_PATH> <(
```

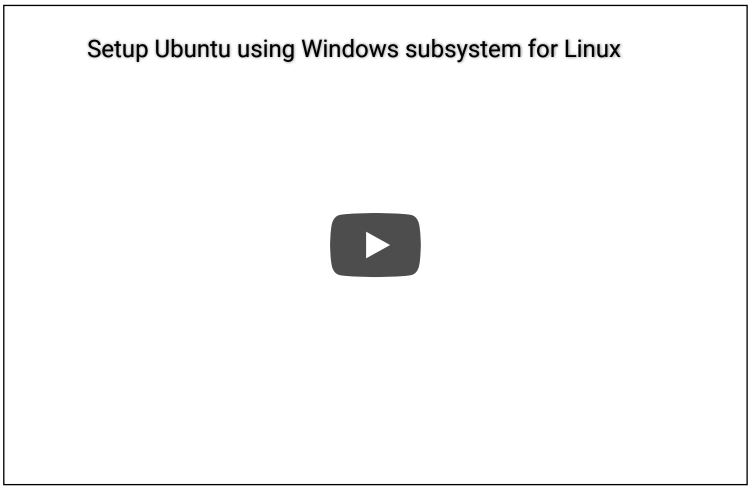




Setup Ubuntu using Windows subsystem for Linux

Now let us see how we can setup Ubuntu on Windows 10

- **Why to setup Ubuntu?**
  - Windows is not completely fool proof in running spark jobs.
  - Using Ubuntu is better alternative and you will run into fewer issues
  - Using Windows subsystem for Linux we can quickly set up Ubuntu virtual machine
- **How to setup Ubuntu using Windows subsystem for Linux?**
  - Follow this link (https://docs.microsoft.com/en-us/windows/wsl/install-win10) to setup Ubuntu using Windows subsystem for Linux
  - Complete the setup process by giving username for the Ubuntu virtual machine



Accessing C Drive using Ubuntu built using Windows subsystem for Linux

- It is better to understand how we can access C drive in Ubuntu built using subsystem for Linux
- It will facilitate us to access files in C drive
- In Linux root file system starts with / and does not have partitions like C drive
- The location of C drive is `/mnt/C`

Setup Java and JDK on Ubuntu

- Before getting started check whether Java and JDK are installed or not
  - Launch command prompt – Go to search bar on windows laptop, type **cmd** and hit enter
  - Type `java -version` If it return version, check whether 1.8 or not. It is better to have 1.8 version. If you have other version, consider uninstall and install 1.8 (Search for programs installed and uninstall Java)
  - Type `javac -version` If it return version, check whether 1.8 or not. It is better to have 1.8 version. If you have other version, consider uninstall and install 1.8 (Search for programs installed and uninstall Java)
  - If you need other versions, make sure environment variables point to 1.8
  - If you do not have Java at all, make sure to follow the instructions and install 1.8 version of JRE and JDK.
- **Why do we need to install Java and JDK?** Scala, Spark and many other technologies require Java and JDK to develop and build the applications. Scala is JVM based programming language.
- **How to install Java and JDK on Ubuntu?**

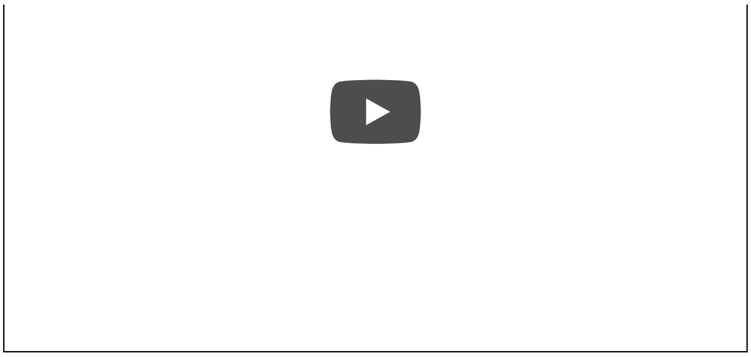
```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

- **How to validate?**
  - Use `java -version` and `javac -version` commands in command prompt and see they return 1.8 or not

Download and Untar Spark

Now let us see the details about setting up Spark on Ubuntu or any Linux flavor or Mac.

- **Why to setup Spark?**
  - Before deploying on the cluster, it is good practice to test the script using spark-submit.
  - To run using spark-submit locally, it is nice to setup Spark on Windows
- **How to setup Spark?**
  - Download spark 2.3 tar ball by going here (https://spark.apache.org/downloads.html). We can use wget to download the tar ball.
    - Choose Spark Release: **2.3.0**
    - Choose a package type: **Pre-built for Hadoop 2.7 or later**
    - It gives the appropriate link pointing to mirror
    - Click on it go to mirror and click on it to download
    - Use tar xzf command to untar and unzip tar ball – `tar xzf spark-2.3.0-bin-hadoop2.7.tgz`
- We need to configure environment variables to run Spark any where



Setup Environment Variables – Mac or Linux

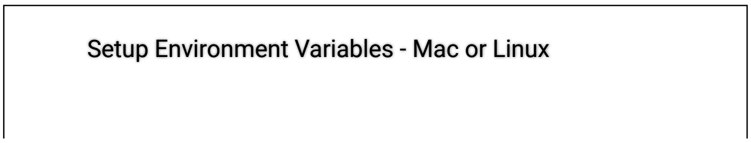
Let us see how we can configure environment variables of Spark

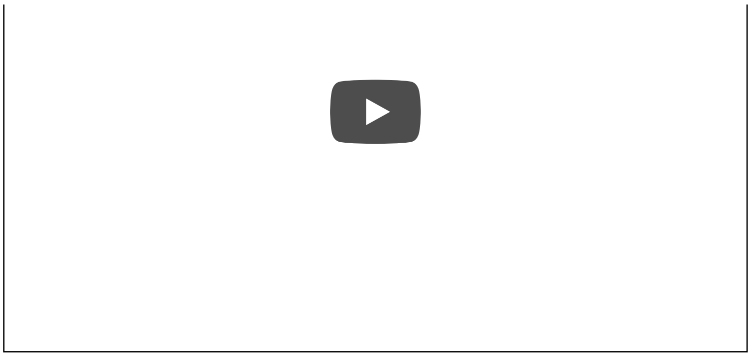
- **Why to setup Environment Variables?** To run spark-submit, spark-shell from any where on the PC using the jar file.
- **How to configure Environment Variables?**
  - Let us assume that Spark is setup under
    - **/Users/itversity/spark-2.3.0-bin-hadoop2.7 on Mac**
    - **/mnt/c/spark-2.3.0-bin-hadoop2.7 on Ubuntu built using Windows subsystem**
  - Setup new environment variable SPARK_HOME and update PATH
  - Make sure to restart terminal (no need to reboot the machine)

```
# On Mac - .bash_profile
export SPARK_HOME=/Users/itversity/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin

# On Ubuntu built using Windows subsystem for Linux - .profile
export SPARK_HOME=/mnt/c/spark-2.3.0-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin
```

- **How to validate?**
  - Go to any directory and run `spark-shell`

```
val orderItems = sc.textFile("C:\\data\\retail_db\\order_items")
val revenuePerOrder = orderItems.
 map(oi => (oi.split(",")(1).toInt, oi.split(",")(4).toFloat)).
 reduceByKey(_ + _).
 map(oi => oi._1 + "," + oi._2)
revenuePerOrder.take(10).foreach(println)
```



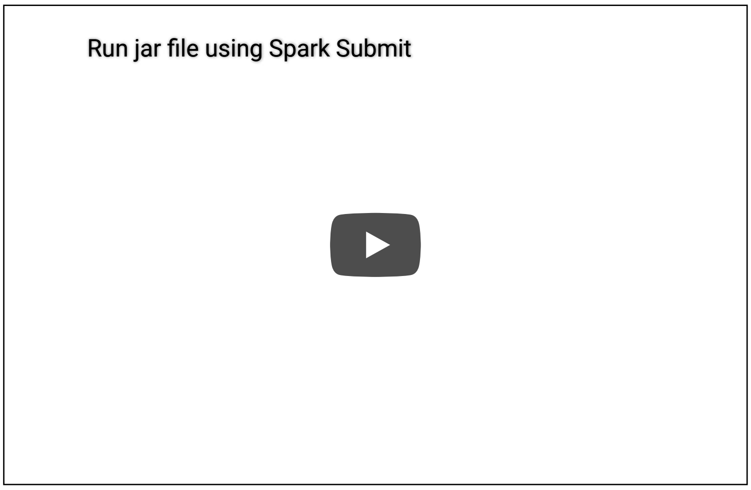


Run jar file using Spark Submit

We can validate the jar file by using spark-submit

- `spark-submit` is the main command to submit the job
- `--class retail_db.GetRevenuePerOrder`, to pass the class name
- By default master is local, if you want to override we can use `--master`
- After spark-submit and control arguments we have to give jar file name followed by arguments

```
spark-submit --class retail_db.GetRevenuePerOrder <PATH_TO_JAR> local <INPUT_PATH> <
```



Conclusion and where to go from here

- This post covers how to set up development environment to work on Spark projects using Scala as a team
- However to gain in-depth knowledge of Spark, you can follow our content and practice on our state of the art big data cluster
- Click here (http://discuss.itversity.com/c/banners) for the coupons for our content

- Click here (https://labs.itversity.com) to go to our state of the art lab to practice Spark hands on for more realistic experience